Memory Management

Beyond Scope

- Variable exists within the scope
  - Disappears outside the scope
  - We want scope-independent objects!

```
Student *createStudent(string name) {
    Student s(name);
    return &s;
}
Student *newStudent;
newStudent = createStudent("kim");
cout << newStudent << endl;
```

←Unexpected result!

C++ way of malloc

- new
  - create an object in heap memory
- delete
  - destroy new'ed object

```
Student *createStudent(string name) {
    return new Student;
}
void removeStudent(Student *s) {
    delete s;
}
```

Memory Leak

- C++ has no garbage collector
  - YOU MUST FREE ALL NEW'ed OBJECTS!
  - Don't forget to delete
- delete X
  - X is a pointer returned by **new** keyword
  - or X is NULL (0) → no effect

Multiple objects

- Array
  - Create an array of objects
    - student_array = new Student[100];
  - Destroy an array of objecgts
    - delete[] student_array;
- Vector
  - Vector is a single object
  - vector<Student> *p = new vector<Student> (100);
  - delete p;

Under the hood

- new, delete, new[], delete[] are implemented using operator functions
  - void *operator new(size_t)
  - void operator delete(void *)
  - void *operator new[] (size_t)
  - void operator delete[] (void *)